BUILD A 2764 BASED "CARTRIDGE" BOARD AND PROGRAMMER FOR YOUR '81

One of the biggest headaches for ZX81/TS1000 users is the painfully slow cassette LOADing performed by their computer. This and the fact that one little glitch during a LOAD can make us have to do it all over again prompted me to design 2764 based "software on eprom" project.

At the time of designing my 2716 based eprom programmer and reader, as previously published in SQ Magazine no. 1, the 2716 was the cheapest per bit eprom on the market. This has now drastically changed. You can currently get a 2764 type eprom for just about the same price as a 2716, yet it contains 4 times the amount of eprom memory (8K x 8 verses 2K x 8) and is faster to boot than the older device. At the prices currently charged for 2764 eproms, it has become feasable to use these devices to store our programs on instead of the cassette tape. And when you consider that once the program has been stored on eprom we can instantly "download" it to ram, PERFECTLY, rather than having to wait on cassettes to LOAD each time, it makes you want to throw your whole cassette library away and put EVERYTHING on eprom cartridges!!

I am not, however, suggesting anything like this. What I am suggesting is storing our most commonly used programs on eprom cartridges and using cassettes for our least used programs. That is the goal of this hardware project.

First, lets cover the specifications of our project. The programmer we will use on our ZX81/TS1000 is actually called by The John Oliger Co. the 2068 EPROM PROGRAMMER. At one time there really was a 2764 programmer specifically designed for the ZX81, but it has since been dropped because it was found that the Oliger 2068 Programmer 1) worked perfectly well on the ZX81/TS1000 for use with 2764 eproms only, 2) was easier to build, and 3) contained a superior circuit than what was used on the older programmer. The only limitation to using the 2068 Programmer on the '81 was that you MUST configure the board for programming only 2764s (IE: the 64-128 switch must be set to "64" for use on the '81) and because the board is physically wider with more edge connector "fingers" than the '81 expansion board, the outermost "2068" fingers MAY have to be ground or filed off the board in order to plug in to the '81 Expansion Board's connectors. These outermost fingers are not used on the 2068 programmer, so it is tryly '81 compatible.

The 2068 programmer set to "64" is physically mapped AT LOCATIONS 2000-3FFFh, on top of any rom/ram you may already have here. You must free up this area before using the programmer by turning off ram using enable switches or removing if a separate "Hunter" type board. If eprom is in this area it can be left aboard as it will not interfere. The programmer contains its own timer and latches, so it is compatible with any dynamic ram memory you might be using. (IE: The Z80 WAIT NOT input line is not used, so refresh of dynamic memory is not a problem with this circuit) You can think of the programmer as "write only" memory, as if you PEEK in this address space you will get either "garbage" or what is in any eprom physically mapped on top of it, but if you POKE in this address range you can write to the eprom in this programmer (If Vpp is at the required voltage).

Programming the eprom is by use of a very simple FOR/NEXT loop from Basic, using the POKE statement with a PAUSE 4 statement following the POKE to allow the timers on the programmer board to time out and complete their cycle. Wnen we

have installed a couple of mc routines permanently on a 2764 eprom (listings enclosed in this booklet), we can then easily program the current Basic program in the computer's memory onto the eprom, via use of a single USR statement. By the way, because this programmer is mapped physically right on top of what COULD be eprom, and to keep things as simple as possible, you cannot verify what is stored on the eprom while it is in the programmer board. You will find that the programmer will VERY rarely ever have an error, so this should not be much of a problem.

A 21VDC regulated supply is required for Vpp on the eprom during programming. (Be sure you ARE using a regular 2764 eprom...the 2764A eprom requires 12.5VDC as its Vpp supply!) I recommend using the Oliger Vpp Power Supply for Vpp, because the EASIEST way to ruin an eprom is give it even a microsecond "glitch" exceeding 26V. The Oliger Vpp Supply has been specially designed to keep this sort of thing from happening. If you would prefer to breadboard this supply instead of ordering it from Oliger Co., send 50 cents to cover the cost of copying and mailing and I will send a copy of this supply's schematic to you. The circuit is very simple and should present no problems in its construction.

The 2764 reader (cartridge) will hold two 2764s and is capable of mapping either of the two eproms anywhere in the 64K map, in 8K blocks. In this project, however, we will map our eproms from C000-FFFFh...the very top of memory. There is also a board enable switch on this cartridge to deselect our eproms. This way we can have more than one cartridge in our expansion board at once, with only the board desired enabled.

Our cartridge is capable of holding more than one program per board. Each program does not have a "name", per se, but is selected by its program number. The first program's number is zero, the second program's number is one, the third's is two, etc. The desired program's number should be POKEd to memory location 16417 before calling the download roitine. The routine's default program is download is program zero, because on power up memory location 16417 contains a zero.

The BASIC programs should be stored on the eprom without any separating bytes, one immediately following the other. The download mc program will calculate where they are and how long they are. All programs are stored on eprom exactly as they are stored on tape, except they are nameless. (IE: From VERSN (4009h) to ELINE-1 (4014/4015h)

The machine language eprom burner program will take care of listable programs if the instructions given in the heading of its listing are followed. This mc program should be stored permanently in memory somewhere in the 2000-3FFFh block on another 2764. Both this program and the downloader program (also listed) are relocatable; You can have them in memory anyplace you like as far as they are concerned. Of course the downloader cannot be held in a Basic REM statement because the downloading itself would wipe it out. (Self destructing software!) Again, the BEST place for both these machine code programs is permanently on 2764 eprom in the 2000-3FFFh block.

For "unlistable" programs, the only way to get them on eprom is to load them in from tape with a program that has data SAVE/LOAD features, such as Sinware's HOT Z. Instructions for doing just this using the original HOT Z V1 are given in another section of this booklet.

Now on to the hardware! In order to use the 2068 Programmer on the ZX81 or TS1000, the T/S rom must be fully decoded to open up this new space. If you have the Oliger ZX81 64K Ram board (SQ#3) this has already been done for you by this board. If you own any other brand of 64K ram, then this would havealready been done too. If you have only 16K of ram and have NOT purchased or bought anything to fully decode the rom, then you can use the 2764 reader board to do this for you. There is sufficient logic on board to decode the rom, but you must remember that every time you use the programmer you must also have this particular reader in the expansion board too. Also, you cannot use the M1 NOT jumper in the reader board that decodes the rom, so it would be best to do it on the board that will contain the 2764 that will be holding the download/ programmer machine code later.

To use the 2764 reader board to decode the T/S rom, simply jumper the "0" output (pin 15) of the '138 chip to the ROM CS NOT edge finger on the board (Sinclair trace no. 23B) with a 1N4148 diode, Cathode (banded end) towards the edge trace. Solder the iode carefully only at the very top of the trace finger and use w/w wire to extend the leads if required. You can also jumper with a w/w wire the "8" output of the '138 to Ea or Eb so that a 2764 can later be mounted on the board in the 8-16K region to hold the machine code programmer/ download routines (physically right on top of the programmer itself). The other eprom socket should be left empty. Now whenever this board is in your expansion board, the T/S rom will be fully decoded. It does not matter if the board's enable switch is off or on.

To map the boards we will actually use to hold our programs from C000-FFFFh, this space must obviously be open for use. If you own the Oliger 64K memory, then all that is needed is to turn off the 48/64 switch on the board. (Now you can see why that switch is on this board!) If you use a Timex/Sinclair 16K ram then you must fully decode it. See SQ #1 for the details on doing this. If you own any other brand of 64K memory, then you must figure out some way to disable the top 16K when desired. There is a circuit shown in this booklet that, although untested, should do the trick for most 64K rams. As you can see, owners of the Oliger 64K ram really have it easy; It's all been done for you!

Now lets actually build our programmers and readers...

OLIGER 2068 PROGRAMMER ASSEMBLY INSTRUCTIONS

Use the assembly instructions provided in the Oliger 2068 Eprom Programmer User's Manual to actually assemble this board. The board is inserted into the Oliger ZX81 Expansion board so that its "slot" aligns with the slot in the expansion board connector. Because this board is wider with more edge fingers than a '81 board, it will hang over a little at both of its ends. This is usually not a problem, but if it is for you simply file or grind off any portion that is hanging over in your way. The circuit itself only uses the "TS1000 compatible" center 23 fingers.

OLIGER ZX81 2764 x 2 CARTRIDGE BOARD ASSEMBLY INSTRUCTIONS

1) Install feedthrough wires in all holes that have round "doughnut" pads on both sides of the board. The exception to this is where the following parts

mount that use the component's leads as the feedthrough, which should be soldered both top and bottom when the part is installed: R1 on both ends, R2 on its top end, and Cbps on its top (+) end. Use #30 stripped wire/wrap wire for the feedthroughs. To install a feedthrough, you simply insert the wire into the hole, bend it over on both sides, solder it on both sides, and then clip it flush on both sides of the board. Remove all traces of flux using a commercial flux stripper, inspect your soldering, and touch up as required.

2) Install the 16 pin IC socket with its pin one end down (towards the edge fingers). Install the two 28 pin IC sockets with their pin one end on the right side of the board. (as viewed from the component side of the board with the edge fingers down) After soldering in place, remove flux and carefully inspect your work.

3) Install the two resistors. Remember to solder R1's leads on both sides of the board and R2's top lead in the same manner. Install the bypass capacitor (Cbps) just right of R2, with its positive lead (usually the longer of the two) in the top hole (and this lead also soldered on both sides of the board).

4) If using this board anywhere in the 32-64K region of memory (as we will for our programs), then jumper pin 6 of the '138 chip to the doughnut labeled "M1". DO NOT install this jumper on the board we will put our machine code routines in that will be mapped 8-16K (or if using this board to decode the ZX rom). If you are using the old Oliger ZX81 printer port, and this board will be mapped at the top of memory, you must also install diodes D1 & D2 wherever you can on the board, as indicated on the schematic. Do NOT install any diodes if you will not be using the Oliger '81 Printer Port or if this board will not be mapped at the very top of memory.

5) If this board is to hold our programs, then jumper pad Epa to the '138's "48" output pad and Epb to its "56" pad. If using the board for eprom in the 8-16K range, jumper one of the Ep pads to the "8" output on the '138. If using this board to decode the ZX rom, install the diode as previously detailed. Install the board enable switch SW1 in its location marked with an "x" between its pads. Plug in the '138 and our cartridge is now ready for use!

HOW TO USE THE PROGRAMMER

Install the eprom to be programmed in the 28 pin socket on the programmer board, with PIN 1 TO THE RIGHT. Power up the computer and the Vpp supply, verify that the Vpp supply is set to "4.4" and "21" and then attach the red and black alligator clips to the programmer's red and black Vpp input wires. Be carefull not to get these clips on the wrong way.

If we wish to put a listable Basic progam on eprom, then we need to load it in from tape. If we desire to put data or machine code on this eprom, then load your mc loader or start POKing it in above ramtop. After all data is somewhere in ram, key in a basic program such as is given below to actually program the eprom.

TYPICAL BASIC PROGRAMMING ROUTINE. Will program 2764 eprom with the contents of ram from C000-DFFFh (49152 to 57343d):

```
10 SLOW
```

```
20 REM X=BASE ADDRESS OF PROGRAMMER
30 LET X=8192
40 FOR N=49152 TO 57343
50 POKE X,PEEK N
60 LET X=X+1
70 PAUSE  4
80 NEXT N
```

After this program is keyed in, or a Basic program is loaded from tape if the mc programming routine is to be used, slide the 4.4/Vpp switch to "Vpp" on the Vpp power supply board. The red Vpp indicator should now light on this board. If you are using the Basic routine, key in: <GOTO 1><ENTER>. If you are using the machine code version key in: <RAND USR XXXX><ENTER> where XXXX equals the memory location of the programming routine (8623 if mapped beginning at 21AFh as in the listing). After <ENTER> is pressed to start the eprom programming, you will need to wait several minutes for the eprom to program. You will know when the eprom is done by the TS report codeat the bottom of the screen. When it IS done, slide the 4.4/Vpp switch back to "4.4", disconnect the Vpp supply from the programmer, and we are done. Your program or data/mc is now stored on the eprom! For a Basic program, if you have the downloader machine code routine on eprom you can now simply mount the eprom you just programmed in another reader board starting at C000h and instantly "LOAD" the program via use of a RAND USR to the location where you have the downloader stored!

FOR HOT Z V1.00 - (LATER VERSIONS MAY USE DIFFERENT KEYS - IDEA IS SAME)

HOW TO PROGRAM A 2764 EPROM USING "HOT Z" (C), TO STORE A ZX-81 PROGRAM IN HIGH MEM (C000H) FOR MC DOWNLOADING.

The program to be stored on eprom must have been previously saved to tape. The eprom will be programmed exactly as it is stored on tape, so if you don't need variables, clear before SAVING to tape to save eprom space.

1) Load or run "HOT Z".
2) Enter C009 in read mode, hit "Y", hit "P", enter FFFF.
3) Hit "J", start tape for Loading, then hit "ENTER".
4) When pattern on CRT screen shows the entire program has been loaded, hit "BREAK". (space)
5) Look at locations C014 + C015.
6) Enter the 2 digets shown in C015, then the 2 digets shown in C014-1. (EG. If C014=8F, and C015=5D, enter 5D8E.)
7) You are now at a location in the 4000-7FFF block. Add 8000h to the number you just entered and enter it. (EG. In our example, you would enter DD8E, because 5D8E+8000=DD8E.)
8) You are now at at location in the C000-FFFF block. Hit "K" to find the decimal equivalent of the address we are at now, and write it down. (EG. In our example, we would write down 56718.)
9) Now hit "Q" to quit to Basic. The 2764 programmer should have already been set up + ready to go, with an eprom in it.
10) Enter a typical Basic eprom programming program. Use 49161 for the first number in the for/next loop, and the number you wrote down for the second number. (EG. In our example, you would have FOR N=49161 TO 56718 for this line.) If the number you previously wrote down is greater than 57352, then you will have to program two eproms to hold the program, as it is longer than 8K. Use 57352 as the second number this time, and use 57353 for the first number of the FOR/NEXT loop next time. You will have to load in the program again at C009 with "HOT Z", and program another eprom with the rest.
11) Turn on Vpp to the programmer, and enter GOTO 1. Your eprom will now be programmed.
12) If there is already another program on the eprom, simply enter LET X=(next avail. bytes' location) for the line setting the programmers' initial starting address. Do not seporate the programs with anything. The downloader program will calculate where the program begins on downloading. Be sure, if you do already have a program on the eprom, that you have enough room left on it to hold the new program. A line like IF X)=16384 THEN STOP in the programming program will stop the routine from going too far.

BASIC PROGRAM PROGRAMMER ROUTINE

This routine will program a 2764 eprom with the current program in memory. The amount of data "SAVED" to eprom will be the same amount as if the normal SAVE command was used, so if you don't need the variables in a program, CLEAR before calling to save eprom space. The USR routine may be called from within a program, if desired, to make the program self running.

POKE 16507 and 16508 either: The location in the eprom you wish the routine to start programming at. (For more than one program on an eprom--default starting address is 2000h) or The location in ram you wish the routine to start from. (In case the program is greater than 8K in length and this is the second eprom to be programmed.) If the last eprom was entirely loaded with the beginning of the current program in memory, then POKE 16507,9 and POKE 16508,96--making the first address to be programmed from this time 6009h. Default source address used by this routine is 4009h. (VERSN) POKE 16507 with the lo byte of the 16 bit address and 16508 with the hi byte. (The same method used when changing ramtop, lo byte first + hi byte last) Be sure both 16507 + 16508 are zero if you are using the default addresses!!!

```
21AF CDE702    STRT CALL 02E7      ;Enter "TEMP FAST" mode
21B2 2A7B40         LD HL,(407B)   ;Get user pass bytes
21B5 AF             XOR A          ;Let 'A'=00
21B6 327B40         LD (407B),A    ;  Clear user
21B9 327C40         LD (407C),A    ;  pass bytes
21BC 7C             LD A,H         ;'A'=hi pass byte
21BD FE20      DFT? CP 20          ;Passed address )1FFF?
21BF 3005           JR NC SRC?     ;Jump for further tests if true
21C1 210940         LD HL,VERSN    ;'HL'=default src
21C4 180A           JR DEST        ;Continue at 21D0
21C6 FE40      SRC? CP 40          ;Address passed )3FFF?
21C8 3006           JR NC DEST     ;Jump if true. Addr. is for src
21CA EB             EX DE,HL       ;'DE'=passed dest
21CB 210940         LD HL,VERSN    ;'HL'=default src
21CE 1803           JR CONT        ;Continue at 21D3
21D0 110020    DEST LD DE,2000     ;'DE'=default dest
21D3 EB        CONT EX DE,HL       ;'DE'=src    'HL'=dest
21D4 E5             PUSH HL        ;Save dest
21D5 2A1440         LD HL,(ELINE)  ;Get last byte to move-1
21D8 A7             AND A          ;Clear carry
21D9 ED52           SBC HL,DE      ;'HL'=byte count
21DB 44             LD B,H         ;  'BC'=byte
21DC 4D             LD C,L         ;  count
21DD EB             EX DE,HL       ;'HL'=src
21DE D1             POP DE         ;Restore dest
21DF EDA0      MBYT LDI            ;Program a byte
21E1 E20702         JP PO SLO?     ;Return via "SLO?" if done
21E4 E5             PUSH HL        ;Save src
21E5 2100C0         LD HL,C000     ;Set 'HL' for test
21E8 19             ADD HL,DE      ;Dest)3FFF?
21E9 3002           JR NC NOER     ;Continue at 21ED if not
21EB CF             RST 08H        ;  Return with report "G"
21EC 0F              REPORT G      ;  Dest)3FFF
21ED D5        NOER PUSH DE        ;Save dest
21EE C5             PUSH BC        ;Save byte count
21EF 210400         LD HL,0004     ;Set 'HL' for the pause
21F2 CD2D02         CALL 022D      ;PAUSE 4
21F5 FD3635FF       LD (IY+35),FF  ;Set FRAMS-hi
21F9 CD4B0F         CALL 0F4B      ;Set DBNC
21FC C1             POP BC         ;Restore byte count
21FD D1             POP DE         ;Restore dest
21FE E1             POP HL         ;Restore src
21FF 18DE           JR MBYT        ;Loop to send another byte
```

EPROM CARTRIDGE DOWNLOAD ROUTINE

This routine will take a basic program stored in the 16K memory block from C000-FFFFh and move it down into the normal Basic ram area. (4009h and on) More than one program may be stored on a board in this block. POKE 16417 the number of the program desired to be downloaded.

The first programs' number is zero, the seconds' is one, etc. so that the routine will default on power up to download the first program in the block. (Because 16417=00 on power uo) The programs should be stored on eprom with no separating bytes. If the desired program, other than program zero, does not exist on the cartridge, the computer will return with report "E", program requested is not on the cartridge board. If there are no programs available, because the cartridge is not inserted into the expansion board, the cartridge board enable switch is off, or the memory board's 48/64K switch is on, the computer will crash.

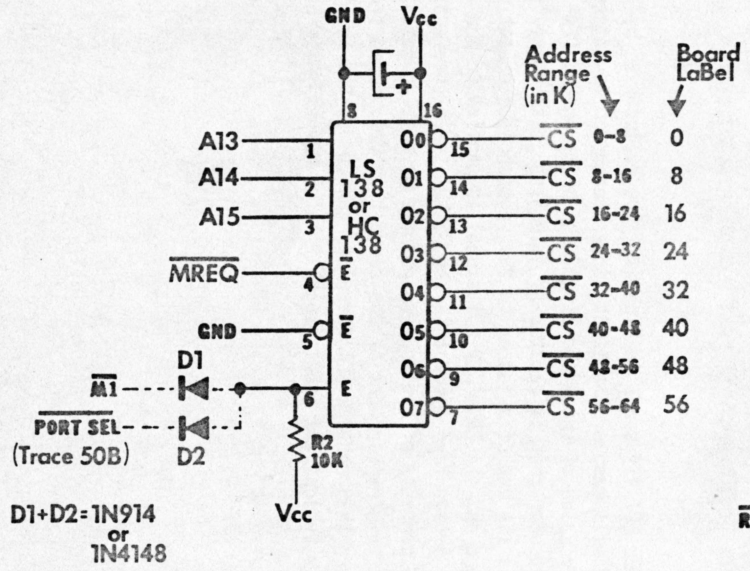
```
2172 CDE702    STRT CALL TFAS      ;Enter "TEMP FAST" mode
2175 2100C0         LD HL,C000     ;Point 'HL' at cartridge slot
2178 110B00    AGIN LD DE,000B     ;Set 'DE' for displcmnt to ELINE
217B 19             ADD HL,DE      ;Point 'HL' at ELINE
217C 5E        EOPA LD E,(HL)      ;  'DE'=ELINE
217D 23             INC HL         ;  address
217E 56             LD D,(HL)      ;
217F 3A2140         LD A,(4021)    ;Get program no. to download
2182 A7             AND A          ;'A'=00?
2183 EB             EX DE,HL       ;'HL'=add ELINE/'DE'=ABS ELINE
2184 2019           JR NZ NEXT     ;Jump if wrong program
2186 010940         LD BC,VERSN    ;Set 'BC' for calculation
2189 C5             PUSH BC        ;Save 4009h on the stack
218A ED42           SBC HL,BC      ;'HL'=byte count
218C 44             LD B,H         ;  'BC'=byte
218D 4D             LD C,L         ;  count
218E EB             EX DE,HL       ;'HL'=ABS address ELINE+1
218F 110C00         LD DE,000C     ;Set 'DE' for calculation
2192 ED52           SBC HL,DE      ;'HL'=start of the program
2194 D1        DEST POP DE         ;'DE'=dest (4009h)
2195 EDB0           LDIR           ;Move the program
2197 E1             POP HL         ;Discard previous return address
2198 217606         LD HL,0676     ;Set HL for new RETurn address
219B E5             PUSH HL        ;Put new return address on stack
219C C30702         JP SLO?        ;Return to 0676h via SLO?
219F 011540    NEXT LD BC,4015     ;Set BC for calculation
21A2 ED42           SBC HL,BC      ;HL=no. of bytes to skip
21A4 19             ADD HL,DE      ;HL=start of next program
21A5 FD3521         DEC (IY+21)    ;Decrement program counter
21A8 7E             LD A,(HL)      ;Get first byte of next program
21A9 FEFF           CP FF          ;Is another program there?
21AB 20CB           JR NZ AGIN     ;Jump if there is
21AD CF             RST 08H        ;  Return w/report "E"
21AE 0D             REPORT E       ;  Program no. not available
```

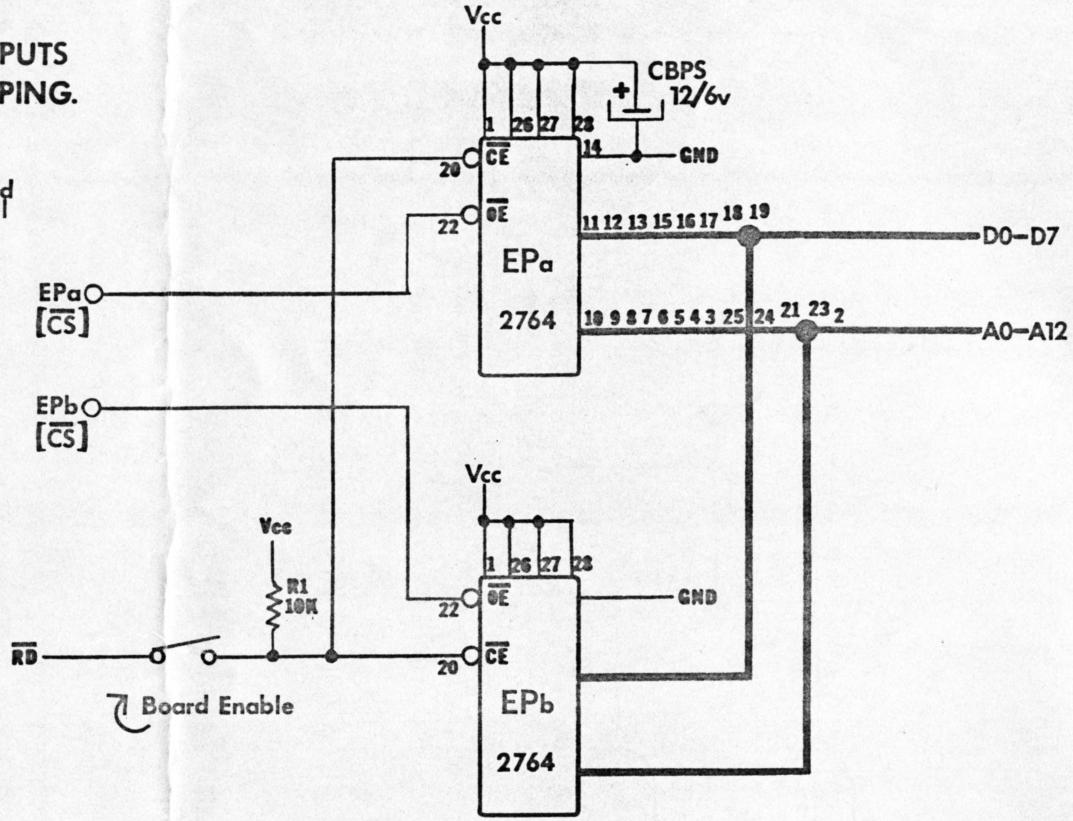
2764 x 2 EPROM READ

FOR INTEL PINOUT TYPE 2764s

Circuit will map 2-2764s ANYWHERE!

JUMPER EP_A + EP_B $\overline{CS}$ TO PROPER OUTPUTS OF THE 138 FOR DESIRED MEMORY MAPPING.





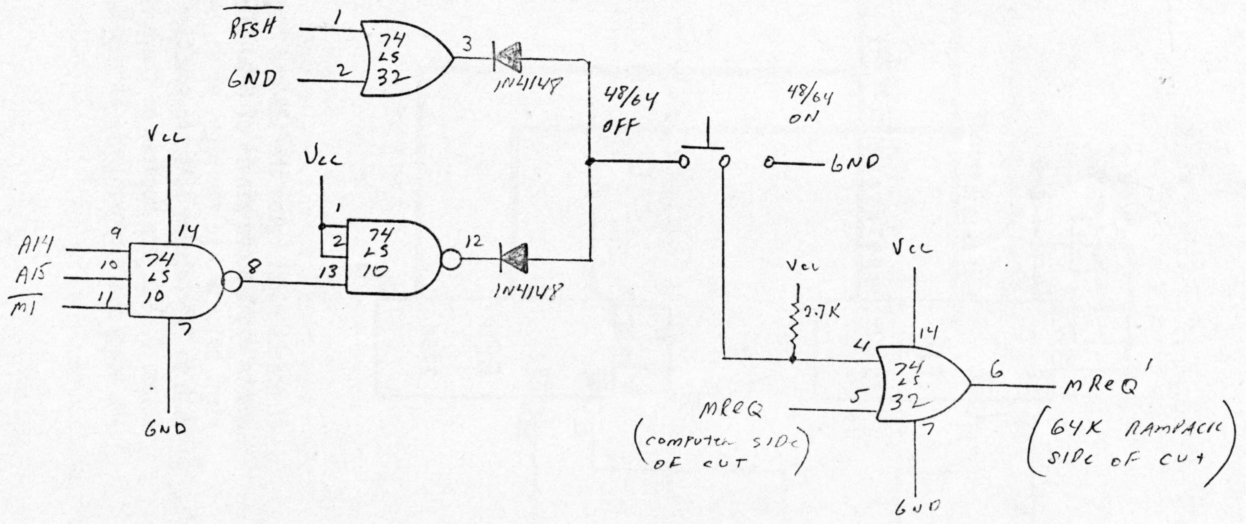
Connect pin 6, above, to M1 NOT if you are using the 48-64K block. If you are using the printer port (FFFFH) and this board mapped 48-64K, then instead of this jumper, connect with D1 + D2 as shown. If this board is mapped anywhere else, R2 will enable the 138 with no additional connections nec.

The 0-8K (0), 16-24K (16), and 24-32K (24) outputs of the 138 cannot readily be used on the ZX-81/TS1000 for obvious reasons. (Ram and Rom monitor)

R1 mounts right below the 138 and Cbps mounts right above pin 14 of EPb. SW1 mounts right below and to the right of R1, marked with an "x" between it's two mounting holes. D1 + D2 are mounted on the bottom of the board where convenient, if needed.

JLO 8308

64 K RAM MODIFICATION TO TURN OFF THE 48-64K BLOCK OF RAM.
(NOT NEC. WITH THE 64K RAM IN SQ NO. 3)



CUT $\overline{MREQ}$ TRACE ON EXPANSION BOARD RIGHT BEFORE RAMPACK, & INSTALL CIRCUIT ABOVE.

2764 READ BOARD Theory of operation

If MREQ NOT is active (low) and PORT SEL NOT (if used) is high (will be pulled high by R2 if not used), the 74LS138 Binary decoder chip will sample it's input lines, connected to addresses 13, 14, and 15, and depending on the binary value of these address lines, bring one of its' output terminals active (low). Thus, the decoder chip is capable of selecting just where in memory a particular eprom is mapped at, via the use of a jumper to its' appropriate output pin.

If the boards' ENABLE switch is switched on, the CPUs' RD NOT signal is applied to the eproms' CS NOT input. Then, if RD NOT goes active, and after this the OE NOT input goes active from the decoder chip, the eprom takes the contents of the address pointed to with address lines A0-A12, and puts it on the data bus. The access requirement of the circuit is determined from the time that RD NET goes active low, rather than when OE NOT goes low, because of the way the eprom is constructed. (rated access time is typically 2 times faster from OE NOT to valid data verses CS NOT to valid data).